Integrating Virtualization, Speed Scaling and Powering On/Off Servers in Data Centers for Energy Efficiency

Julian A. Gallego Arrubla, Young Myoung Ko, Ronny J. Polansky, Eduardo Pérez, Lewis Ntaimo and Natarajan Gautam*

Department of Industrial and Systems Engineering
Texas A&M University
4012 ETB, College Station, TX 77843-3131

Abstract

Data centers consume a phenomenal amount of energy which can be significantly reduced by appropriately allocating resources using technologies such as virtualization, speed scaling and powering off servers. We propose a unified methodology that combines these technologies under a single framework to efficiently operate data centers. In particular, we formulate a large-scale mixed-integer program (MIP) that prescribes optimal allocation of resources while incorporating inherent variability and uncertainty of workload experienced by the data center. However, only for small to medium-sized clients it is possible to solve the MIP using commercial optimization software packages in a reasonable time. Thus for large sized clients we develop a heuristic method that is effective and fast. We perform an extensive set of numerical experiments to illustrate our methodology, obtain insights on the allocation policies, evaluate the quality of our heuristic, and test the validity of the assumptions made in the literature. The results show that gains of up to 40% can be obtained by using the integrated approach over the traditional approach where virtualization, dynamic voltage/frequency scaling and powering off servers are done separately.

1 Introduction

Data centers are among the fastest growing enterprises in the U.S. economy. Growing demand for data services, the availability of high-volume Internet gateways, and the relatively modest facilities requirements for server banks have led to an explosive growth in the data center industry. Practically every single organization, especially if they have a web page, whether a private company or a public undertaking, uses the services of a data center (in-house or outsourced, usually the latter) to acquire, analyze, process, store, retrieve, and disseminate information. While these data centers provide high quality of service, they also consume a phenomenal amount of energy. In a year, data centers use about 60 billion kilowatt hours of electricity, accounting for 1.5% of all consumption in the United States. Industry-wide data centers currently spend over $5 Billion annually on electricity [10] and for their growth alone 10 new power plants would be needed by 2013 [27].

*Corresponding Author: gautam@tamu.edu

Further, the greenhouse gas emissions by data centers is next to airlines, shipyards, and steel plants. The combined greenhouse gas emissions from all data centers exceed what entire countries like Argentina or the Netherlands emit [7]. Therefore, reducing energy consumption and thereby greenhouse gas emissions in data centers is of paramount importance from an environmental standpoint. In addition to the environmental impact, reducing energy consumption would also result in serious economical gains. Computerworld [21] reports that power-related issues are already a top concern in all data centers. For every watt of power used by IT equipment in data centers today, another watt or more is typically expended to remove waste heat [21, 23]. In fact, energy costs are higher than the cost to lease the space for data centers [19].

The key problem is that most of the energy consumed by data centers is not for useful work. The utilization of an average server is very low, but it also consumes energy and generates heat while being idle. One of the reasons for this low server-utilization is that most data centers are built up of cheap servers running a single application [17]. A few data centers have begun to remove dead servers, enable power-save features, and power off servers when not in use. But there are other strategies that can be explored. IBM [11] recently announced that it should be possible to achieve about 40-80% energy savings in data centers by power management. According to [8], "if you were to implement all the data center efficiencies that could be reasonably achieved by 2015, it would save the equivalent of the annual electricity consumption of 1.8 million homes." To achieve that much of energy savings, there exist technologies such as virtualization and dynamic voltage frequency scaling (DVFS) that enable energy conservation. Virtualization allows data centers to consolidate applications on a server. DVFS enables servers to run at slower speeds thereby significantly lowering energy consumption. Hence it is also referred to as *speed scaling*. In addition, depending on the load experienced, one could power down servers from time to time.

In this paper we devise a unified approach for resource allocation in data centers to remove all feasible inefficiencies by combining virtualization, DVFS and powering on/off servers. The state of the practice is that only a few data centers have begun considering these technologies and those that do, use a somewhat ad hoc approach. However, the state of the art in terms of research is mature and is summarized in Section 2. One of the major shortcomings of the previous research is that it separates the *strategic* problem of virtualization (i.e., mapping applications to servers) from the *operational* problem of DVFS and powering on/off. In particular, the articles on the *operational problem* assume a certain structure for the *strategic* problem which we show is not necessarily optimal. Likewise, the articles on the strategic problem are not operation-aware which also is not optimal.

To address the key energy-efficiency issues in data centers, we leverage upon the benefits of virtualization, DVFS and powering servers on/off. We formulate the problem as a mixed integer (0-1) program (MIP) that considers all three techniques and implicitly provides quality of service (QoS). Our formulation determines which applications to allocate to which servers. Given a load profile for various applications in a data center, our model decides how best to allocate them to various servers via virtualization so that energy cost of operations is minimized. We determine power on/off decisions, request-routing allocations and adjust the frequency of each server at a fine granularity. However, this MIP is a large scale problem that is difficult to solve directly. Therefore, we devise a heuristic method that provides close to optimal results for large instances of the problem. We also compare the results of our integrated approach to the traditional case where virtualization, DVFS and powering on/off servers is done separately. The results show that the integrated approach provides gains of up to 40% in terms of energy consumption.

The contributions of this work are: a) a unified methodology that integrates virtualization, DVFS, request routing and powering on/off servers so that the strategic decisions are operation-aware and vice-versa; b) an MIP formulation of the integrated energy consumption reduction problem; c) a heuristic method for solving the problem; and d) extensive computational results that demonstrate that using the integrated approach over the traditional approach where virtualization, dynamic voltage/frequency scaling and powering off servers are done separately provides substantial energy savings. The rest of this paper is organized as follows: In the next section we review related work and in Section 3 we state the relevant research questions. We give a formal problem description and derive the formulation in Section 4 along with a description of how we resolve uncertainty and non-stationarity while providing QoS. In Section 5 we derive a heuristic solution method for the integrated problem and a two-phase heuristic for the non-integrated problem. We report computational results in Section 6 and end with some concluding remarks and ideas for future work in Section 7.

2 Literature Review

The recent rise in data-center energy costs has motivated the research that aims to reduce power consumption (viz. using dynamic voltage/frequency scaling (DVFS), strategically powering on/off servers, routing and virtualization). Speed scaling or DVFS enables servers to run at slower processor speeds thereby significantly lowering energy consumption. Horvath et al. [14] present a nonlinear optimization approach to control the energy efficiency of a multi-tier web server using

a Dynamic Voltage Scaling (DVS) scheme. While acknowledging that turning servers on and off can contribute to a reduction in the energy usage of multi-tier web servers, the authors assume that such an energy savings approach has previously been implemented and consider the case where all machines are on. Herbert and Marculescu [13] examine the benefits of using DVFS on chip-multiprocessors. The authors explain the advantage of using DVFS and partitioning the processor into voltage/frequency islands which consist of clusters of cores that are running at the same frequency.

DVFS can be combined with turning the servers on/off. Petrucci et al. [25] study such a problem and optimize power consumption levels by formulating an MIP. For large problem sizes ranging from 150 to 500 servers, the solution approach presented was unable to find an optimal solution within a prescribed time (300 seconds), so the authors present the solution gap between the best feasible solution found and the lower bound. The key assumption is that all applications reside in all servers (we call that a *cluster* of servers). Chen et al. [5] investigate the use of multiple power management techniques in data centers. Their model incorporates turning servers on/off and DVFS with all servers (in a cluster) running at the same frequency, while providing a QoS guarantee. Three different approaches for the power efficiency problem are investigated; a pro-active approach, a reactive approach, and a hybrid approach that combines the previous two. The pro-active model predicts the incoming workload behavior and uses this information in a queueing model to control the power usage of the system. The reactive model controls the power management strategy by using feedback control, while the hybrid scheme combines the two approaches by using the predictive information gathered to allocate applications to servers and the feedback to control the DVFS scheme. Bertini et al. [1] develop a MIP model for the energy-efficiency problem for data centers. Energy saving strategies incorporated in their model include turning servers on and off and choosing what speeds the servers will run at. A QoS constraint is also employed to control the fraction of deadlines met and by how late the requests are upon completion.

Having described DVFS and powering on/off, next we consider virtualization. Virtualization allows data centers to run more than one application on a server. Kusic et al. [16] use virtualization and examine how turning servers on and off can reduce the power consumed by the physical machines. The authors formulate the problem of determining the allocation of resources as an optimization problem. Control theory is used to handle adapting to the varying workload, while attempting to satisfy a QoS constraint. To validate the setup, the authors simulate a system consisting of six servers with two applications. Padala et al. [22] take a control theoretic approach to achieve high utilization of servers in data centers by adopting virtualization techniques while

meeting QoS goals. They examine the case when the system consists of multi-tier applications with varying QoS goals at each tier and conduct their experiments using five servers, two of which host two applications. This is in contrast with our objective since we seek to maximize the energy efficiency of the system while meeting QoS goals. Also, DVFS and turning servers on and off are not implemented. Wang et al. [31] take a two-layer control theoretic approach to the energy efficiency problem of virtualized systems. On one level, the strategy is to attempt to balance the load among the virtualized machines so that they can have similar response times. On the other level, the frequency of the CPU is adjusted in order to increase the power efficiency. Their setup uses two computers that hosts three virtual machines. However, the configuration does not account for the ability to turn servers on and off.

The process of assigning applications to servers has been studied using different techniques that include graph-theoretic approaches as well as heuristics. For example, Houstis [15] determines an optimal allocation of an application to a k-processor system with the objective of minimizing the total processing time for an application. More recent articles on assigning applications to servers include Chandra et al. [2] and Urgaonkar et al. [29]. Chu and Lan [6] propose a heuristic to allocate tasks to processors in a distributed system. They also assume that tasks have been partitioned into modules that are to be allocated to processors. Chen et al. [3] addresses issues with load balancing and proposes routing strategies that enable powering on-off of servers. Le et al. [18] consider routing, not within a data center, but across geographically distributed data centers so that cost is minimized with minimal environmental impact. Tindell et al. [28] use simulated annealing to address the task assignment problem for a set of processors that have a fixed processing speed and capacity. Garcia and Harbour [12] show how heuristics can be used to assign priorities to tasks in a distributed real-time system to minimize end-to-end delay.

There has also been significant research proposing new architectural designs for data centers to optimize energy-efficiency as well as research on understanding which data center components contribute the most to power-efficiency issues. Meisner et al. [20] propose an energy saving approach called PowerNap. The proposed design seeks to minimize the amount of power consumed while the system is operating in a state with a low workload while also minimizing the time required to transition from a high power state to a low power state, and vice versa. The design works by alternating between a high power state and a low power state. The authors compare their concept with a system capable of utilizing DVFS by comparing performance metrics such as energy savings and response time based on a queueing model.

Pelley et al. [24] develop models that can be used to simulate the major components of a

data center including servers, power conditioning equipment, cooling systems, networking systems, and lighting systems. The authors show which subsystems of a typical data center consume the most power. As expected, servers typically consume the most power in data centers and the authors simulation model validate this assumption. In fact, the authors show that servers typically consume 56% of the total power utilized. To put this into perspective, the next highest power-consuming subsystem in a data center is the cooling system which consumes 30% of the available power, while the remaining subsystems consume the remaining 14% of the available power. That said, Table 1 summarizes the power management techniques used in papers that are most related to our research.

Table 1: Summary of related work using multiple servers and multiple applications

Paper	Assigning Applications To Servers	Routing Requests Among Servers	DVFS	On/Off	Primary Methodology
[1]	No	No	Yes	Yes	MIP
[3]	No	Yes	No	Yes	NLP/Q
[5]	No	No	Yes	Yes	NLP/Q
[14]	No	No	Yes	No	NLP
[16]	No	No	No	Yes	CT
[25]	No	No	Yes	Yes	MIP
[29]	Yes	No	No	No	MIP
This work	Yes	Yes	Yes	Yes	MIP

Legend: MIP - Mixed Integer Programming; CT - Control Theory; NLP - Nonlinear Programming; Q - Queueing

3 Research Questions

Having reviewed the literature, we now postulate research questions that we seek to answer by way of this study. Most of the research questions are based on the shortcomings of the literature. In particular, the strategic problem of allocating applications to servers, and the operational problem of deciding the server speeds, routing and power on/off have been dealt with separately in the literature. We will study whether they need to be considered in a unified framework. In particular, we would like to know if the assumptions made while studying them separately are reasonable when they are considered in a unified framework. That motivates us to ask the following key research questions:

1. *Are servers clustered across applications?* All relevant papers in the literature, which consider DVFS and/or powering on/off servers such as Petrucci et al. [25] and Chen et al [5], not only assume that the allocation of applications to servers is already done, but also resulting in clusters of servers with the same set of applications. For example, three servers would all contain the same four-application set $\{a_1, a_2, a_3, a_4\}$ for some $a_i \in A$ $(i = 1, 2, 3, 4)$. The

rationale behind this assumption is that servers can be powered down during lean periods and fired up at fastest frequencies during peak periods.

2. *Are applications positively or negatively correlated assigned to the same server?* A natural approach is to pair up applications that are positively or negatively correlated and run them on a single server (see Verma et al. [30]). The rationale behind that is that when applications are positively correlated, during lean periods the servers can be powered down, whereas when they are negatively correlated the effective load is lower and thereby one can operate at a lower server frequency. Clearly, it would enable decoupling the strategic decision of assigning applications to servers from the operational ones.

3. *Are assignments based on peak loads a good idea?* Given a fixed capacity of a data center, it is fairly common (see Peak Clustering based Placement in Verma et al. [30] and references therein) in resource allocation to make decisions only considering peak periods. Then, it is also common to use this resource assignment as a benchmark for other periods including lean periods. The notion is that if we carefully allocate resources during peak loads, one can always design methods to reduce energy consumption during lean periods. The argument that goes against this is that during peak loads one does not have much of a choice but run the system at full capacity, it is only during lean periods there is scope to save energy and this must be done effectively.

4. *Are servers running at the same frequency in a given time interval?* While solving the stochastic control problem of deciding what frequencies or speeds to run servers, it is often convenient (as done in Chen et al [5]) from an analytical standpoint that all "on" servers run at the same frequency at any given time. The rationale is that if the set of frequencies is from a continuous spectrum, then the frequency used by every server would converge to a constant value.

5. *Is the workload for each application split across servers similarly over time?* DVFS and powering on/off are sometimes performed independently of each other at each server with a forecast of workload such that workload is split across servers in a pre-assigned manner (see Chen et al. [3]). This means that the workload is more-or-less a constant across time for all applications and servers.

6. *Is it sufficient to just consider powering on/off without doing DVFS?* Articles in the literature (Meisner et al. [20]) have suggested that it may be efficient to just power servers on and off

without worrying about DVFS. So if a server is on, it just runs at the maximum frequency or it is powered off. The intuition is that a server might as well run at highest speed and provide excellent QoS since even at the lowest speed a significant amount of power is consumed.

7. *Is the selected frequency during any interval of bang-bang nature, i.e. the highest or lowest possible frequency?* Many articles (viz. Dhiman et al. [9]) suggest that instead of having a spectrum of frequencies it is sufficient to just have two, the highest and lowest possible. This way the server could pile up some requests, then serve them all at the fastest rate. Although our formulation does not consider changing frequencies at such a small granularity, a bang-bang policy at the larger granularity could certainly imply the same in the smaller granularity.

8. *Is it necessary to consider all the applications while making assignment, routing and frequency selection?* Data centers like many other systems experience loads from applications based on the Pareto principle, i.e. 20% of the applications (called elephants) constitute 80% of the load. Why not just use those 20% of applications, i.e., *elephants* (as described in Singla et al [26]), to make decisions, perhaps the problems would be faster to solve? A simpler question to ask is whether allocation based on the *elephants* will provide a near-optimal allocation when *mice* (the remaining 80% of applications) are also present.

Next we present our methodology and provide answers to the above research questions later in Section 6 after presenting our computational results.

4 Problem Description and Model Formulation

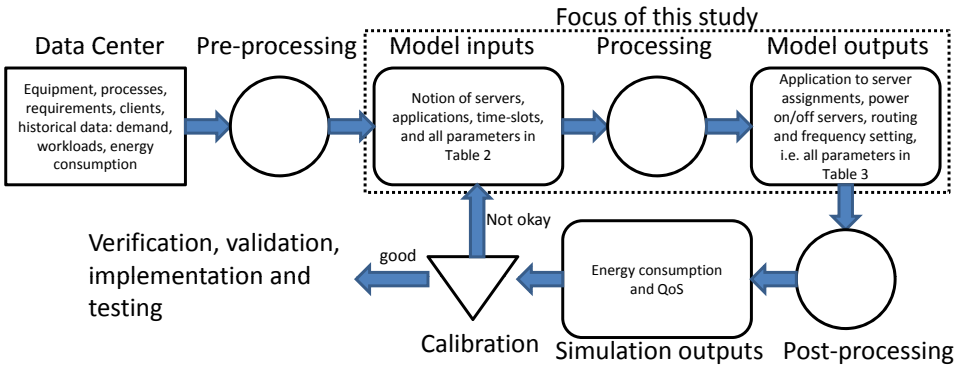

Figure 1: Process of modeling, analysis and implementation

We first describe the scope of this research study depicted in Figure 1. In this paper we focus on the *Processing* step that takes the *Model inputs* and converts them to *Model outputs*. However, we assume that the pre-processing step of converting the data center system into the "model inputs" is already done. Understandably, this is an art which would be tailored to individual data centers. In particular, the data center would have to model equipment (by aggregating or decomposing) abstractly so that they can be called servers. Likewise client processes and requirements would have to be mapped to applications. Then based on historical data, model-parameters such as demand, workload and energy consumption over time and space can be obtained and an appropriate way to partition the decision horizon into time-slots would be determined. This would result in the parameters that we later list in Table 2. Another crucial aspect that is left out of this study includes the post processing phase of determining the true energy consumption and QoS by running simulations as well as issues such as verification, validation, implementation and testing. Once again, these would be specific to individual data centers and would have to be done on a case-by-case basis. That said, we next describe the detailed model and formulation, followed by some theoretical underpinnings.

4.1 Model and Formulation

We consider a data center with a set of applications A that need to run on a set of (not necessarily homogeneous) servers S. These applications can be web services, databases, etc., and they are CPU intensive. More than one application can run on a server at a given time using virtualization technology (if necessary). In addition, it is crucial to note that an application can simultaneously run on more than one server. Thus servers run multiple applications and an ingress router sends incoming requests to the servers running the appropriate application. We seek to answer the strategic question of how to assign applications to servers bearing in mind that not more than C_j applications can be assigned to server j for all $j \in S$. By assigning applications to servers we mean using a single physical server to run multiple virtual servers, one virtual server for each application. So this virtual server set up is one time and it is possible for jobs to migrate from one virtual server to another in small time scales which we take advantage of between time intervals.

Once the strategic assignments are made, the operational decisions include whether to power on or off a given server, if on at what speed to run the server and what fraction of an application's requests to route there. The technology that enables a server to run at many speeds is DVFS. Essentially a server's CPU can be slowed down by adjusting the voltage and/or frequency thereby significantly reducing the energy consumed. The frequency of a CPU is directly proportional to

the voltage supplied, and for the remainder of this paper, we shall use the term *frequency* instead of voltage (in fact in the literature this is also referred to as DVFS for Dynamic Voltage/Frequency Scaling or just speed scaling). Measurements have shown that the energy consumption is approximately a third-order polynomial function of the frequency (see references in Chen et al [5]). However, slower frequency does mean that time to process jobs would be correspondingly slower too.

We consider a planning horizon that encompasses a set of time slots T. In each time slot we decide whether each server must be powered on or off and if on, at what frequency it must run in that time-interval (the application to server assignment remains fixed during the entire planning horizon). For example, in all our numerical experiments we consider a planning horizon of one day and each day is divided into 24 one-hour time slots within which the server runs at a particular frequency (with a frequency of zero corresponding to the server powered off). We assume that a forecasted average load profile is available for each application during each time slot. Note that in many cases the forecast load remains relatively unchanged from day to day resulting in identical application to server assignments on a daily basis.

For convenience, we provide the notation we use to describe our formulation in Tables 2 and 3. In our model λ_{it} will denote the forecasted average job arrival rate for application i at time slot t (in units of number of jobs per second) for all $i \in A$ and $t \in T$. Also, $1/\mu_i$ is the average workload that each arriving class i (for all $i \in A$) brings in for service. The speed at which jobs can be processed by a server is proportional to the frequency the server's CPU runs at. Specifically, let Z_j be the set of possible frequencies for server j using DVFS (for all $j \in S$). Then for any $j \in S$, the speed of server j in terms of the amount of workload it can process per second is given by γ_{jz} for all $z \in Z_j$. To provide reasonable QoS for the job requests, we use a maximum allowable traffic intensity at each server to be ρ. In other words, the average workload that arrives at server j per unit time for all $j \in S$ must not exceed $\rho\gamma_{jz}$ at any time period if the server runs at frequency $z \in Z_j$. We will show in Section 4.2 how this addresses uncertainty, non-stationarity and QoS. Also, the data center incurs a cost of β_{jzt} per second when server j is operated at frequency z during time interval t for all $j \in S$, $z \in Z_j$ and $t \in T$.

The inputs to the decision-making framework considered here are: A, S and T, as well as variables based on them such as Z_j, C_j, γ_{jz}, λ_{it}, μ_i, ρ and β_{jzt} for all $i \in A$, $j \in S$, $z \in Z_j$ and $t \in T$. The goal of this paper is to effectively manage resources through virtualization, DVFS, routing and powering on/off by considering them in a single framework. Our objective is to minimize the energy cost incurred to operate the data center during the course of a planning horizon so that

Table 2: Notation corresponding to problem statement inputs

Sets	
A	index set of applications
S	index set of servers
T	index set of time slots (e.g. every hour of a week)
Parameters	
Z_j	index set of frequency options of server $j \in S$
C_j	maximum number of applications assignable to server $j \in S$
γ_{jz}	capacity of server $j \in S$ (workload per second) under frequency $z \in Z_j$, units: workload/time
λ_{it}	average number of arrivals per second for application i at time slot t for all $i \in A$, $t \in T$
$1/\mu_i$	average workload brought by each arriving application i request for all $i \in A$
ρ	target load for all servers (surrogate for quality of service)
β_{jzt}	cost incurred per second when server j runs at frequency z at time t, $\forall\ j \in S$, $z \in Z_j$, $t \in T$

Table 3: Decision variable list $\forall i \in A, \forall j \in S, \forall z \in Z_j, \forall t \in T$

Decision Variables	
x_{ij}	$x_{ij} = 1$ if application i is assigned to server j, 0 otherwise
v_{ijt}	fraction of arrivals of application i assigned to server j during interval t
u_{jzt}	$u_{jzt} = 1$ if server j runs at frequency z during time t, 0 otherwise

capacity and QoS needs are met. To do that we need to decide how applications are to be allocated to servers (virtualization), which server to route each request for various applications and what frequencies the servers should run (DVFS and power on/off) at each time period.

Let x_{ij} be the corresponding decision-variable so that if $x_{ij} = 1$, application i is assigned to server j, otherwise $x_{ij} = 0$ for all $i \in A$ and $j \in S$. Recall that an application can be assigned to more than one server, and a server can run more than one application. Note that this assignment does not change over the entire planning horizon. However, what could change during each time slot in the horizon is the traffic split for various applications to servers they are running on. In particular, let v_{ijt} be the fraction of arrivals of application i that are assigned to server j during time interval t ($\forall\ i \in A$, $j \in S$ and $t \in T$). The final set of decision-variables deals with DVFS and powering on/off servers during each time interval $t \in T$. For server j (for any $j \in S$) recall that Z_j is the available set of frequency indices. We assume that this set also contains a frequency $z = 0$ corresponding to the server powered down. Thus, we define u_{jzt} for all $j \in S$, $z \in Z_j$ and $t \in T$ as $u_{jzt} = 1$ if server j is running at frequency z during time interval t, otherwise $u_{jzt} = 0$.

To minimize the cost incurred during a planning horizon subject to satisfying various capacity, feasibility and allocation constraints we propose the following formulation (with decision variables

x_{ij}, u_{jzt} and v_{ijt} for all $i \in A$, $j \in S$, $z \in Z_j$ and $t \in T$):

$$\text{MIP1} : \text{Min} \sum_{j \in S} \sum_{z \in Z_j} \sum_{t \in T} \beta_{jzt} u_{jzt} \tag{1a}$$

$$\text{s.t.} \sum_{z \in Z_j} u_{jzt} = 1 \quad \forall j \in S, \forall t \in T \tag{1b}$$

$$\sum_{j \in S} v_{ijt} = 1 \quad \forall i \in A, \forall t \in T \tag{1c}$$

$$\rho \sum_{z \in Z_j} u_{jzt} \gamma_{jz} \geq \sum_{i \in A} \frac{\lambda_{it}}{\mu_i} v_{ijt} \quad \forall j \in S, \forall t \in T \tag{1d}$$

$$\sum_{i \in A} x_{ij} \leq C_j \quad \forall j \in S \tag{1e}$$

$$v_{ijt} \leq x_{ij} \quad \forall i \in A, \forall j \in S, \forall t \in T \tag{1f}$$

$$\sum_{i \in A} v_{ijt} \leq C_j (1 - u_{j0t}) \quad \forall j \in S, \forall t \in T \tag{1g}$$

$$v_{ijt} \geq 0, x_{ij} \in \{0,1\}, u_{jzt} \in \{0,1\} \quad \forall i \in A, \forall j \in S, \forall z \in Z_j, \forall t \in T \tag{1h}$$

The model MIP1 aims to minimize the total cost incurred during the planning horizon. Since u_{jzt} is a binary variable denoting whether or not server j used frequency z at time t, we can write down the associated cost for server j during interval t as the sum over all z the product $\beta_{jzt} u_{jzt}$. Thus by summing that over all servers across the planning horizon we get the objective function (1a). Constraint (1b) ensures that during an interval t, server j can run only at one frequency (including $z = 0$ denoting a server being off). Thus if server j runs at frequency z at time t, then $u_{jzt} = 1$ and $u_{jz't} = 0$ for all $z' \neq z$. Constraint (1c) ensures that all of application i traffic is divided across various servers at every time interval. Since v_{ijt} is the fraction of application i requests in time interval t that gets assigned to server j, the sum over all servers must be one.

Next, recall that the average workload that arrives at server j must not exceed $\rho \gamma_{jz}$ at any time period if the server runs at frequency z. By selecting a $\rho < 1$ using the process described in Section 4.2, we ensure that the QoS is met under a stochastic and time-varying environment. Through constraint (1d) where the LHS is the assigned capacity of server j in time interval t and the RHS is the total workload that arrives to server j in time interval t we ensure that the mean workload be no more than ρ times the assigned service capacity. In Section 4.2, we describe the theoretical underpinnings to justify that constraint (1d) would account for uncertainty, variability and QoS. Next, notice that x_{ij} determines whether or not application i is assigned to server j (during the entire course of the planning horizon). Constraint (1e) enforces that not more than C_j applications are assigned to server j. Constraint (1f) ensures that if application i is not assigned to server j, then no fraction of arriving requests be assigned to that server during any interval of time.

Further, it is critical that if server j is off during time t, then no traffic (across all applications) be assigned to that server (that leads to Constraint (1g)). Although Constraints (1d), (1e) and (1f) imply Constraint (1g), we added Constraint (1g) to tighten the formulation. Constraint (1h) ensures non-negativity and binary nature of the decision variables.

In the above formulation, we would like to determine the number of decision variables and number of constraints. For that we make a simplifying assumption that $Z_j = Z$ for all j (i.e. the set of available frequencies are the same for all servers). Then there are a total of $|S||A||T|$ real decision variables and $|S|(|A| + |Z||T|)$ binary decision variables. Also, the number of constraints are $3|S||T| + |A||T| + |S| + |A||S||T|$, not including the non-negative and binary constraints. It may be worthwhile getting a perspective of the problem dimension. If we first consider a 1-day planning horizon with $|T| = 24$ hourly intervals and all servers with 8 non-zero frequencies, then including the frequency of zero corresponding to the off state, $|Z| = 9$. For example, for small-sized clients with 20 applications and 10 servers, we have 4,800 real decision variables, 2,360 binary decision variables, and 5,810 constraints (not including binary and non-negativity constraints). Also, for medium-sized clients with 40 applications and 20 servers, there would be 19,200 real decision variables, 5,120 binary decision variables, and 21,220 constraints (not including binary and non-negativity constraints). Thus for large-sized clients the mathematical program becomes extremely huge.

4.2 Theoretical Foundations Resulting in Constraint (1d)

In this sub-section we use fluid and diffusion limits to describe the methodology to obtain ρ in constraint (1d) so that it would account for uncertainty, variability and QoS. For details on the fluid and diffusion scaling, refer to Whitt [32] or Chen and Yao [4]. We first describe the results in an abstract manner and towards the end of the section, we tie them to our data center context. Consider a $G/G/1$ queueing system indexed by a scaling factor n with arrival rate λ_n and squared coefficient of variation (SCOV) of inter-arrival times C_a^2. Each arrival brings a random amount of work which is independent and identically distributed (IID) with mean $1/\mu$ and SCOV C_s^2. Let f_n be the amount of work that can be processed per unit time, i.e. frequency or speed. Therefore, the service times have a mean $\frac{1}{\mu f_n}$ and SCOV C_s^2. We assume that $0 < \lambda_n < \infty$, $0 < f_n < \infty$, $C_a^2 < \infty$, $C_s^2 < \infty$. Also, as $n \to \infty$, $\lambda_n \to \lambda$ and $f_n \to f$ such that $(1 - \frac{\lambda_n}{\mu f_n})\sqrt{n} \to \zeta$ such that $0 \leq \zeta < \infty$.

Define the virtual delay at time t by $V(t)$ so that if an arrival occurs at time t, it would experience a random delay $V(t)$ to begin service under a first-in-first-out queue discipline. We assume that $V(0) = 0$, which implies an initially empty system (this assumption can be relaxed but we retain it because it would not affect our eventual analysis). Using Functional Central Limit

Theorem (see Whitt [32]) it can be shown that the stochastic process $\{\frac{V(nt)}{\sqrt{n}}, t \geq 0\}$ converges to a Brownian motion with drift $-\zeta$, variance term $\frac{\lambda}{(\mu f)^2}(C_a^2 + C_s^2)$ and a barrier at the origin. Such a *reflected* Brownian motion converges to a stationary distribution which is exponential with mean $\frac{\lambda(C_a^2+C_s^2)}{2\zeta(\mu f)^2}$. Therefore for some positive constant α, we have

$$\lim_{n\to\infty} P\{V(nt) > \alpha\sqrt{n}\} = \lim_{n\to\infty} e^{-\frac{2\alpha\sqrt{n}(1-\lambda_n/(\mu f_n))(\mu f)^2}{\lambda(C_a^2+C_s^2)}}.$$

Using $\alpha\sqrt{n} = \delta$ we have

$$\lim_{n\to\infty} P\{V(nt) > \delta\} = e^{-\frac{2\delta(1-\lambda/(\mu f))(\mu f)^2}{\lambda(C_a^2+C_s^2)}}.$$

Now consider a single server in a data center that is powered on during a particular one-hour interval. Arrivals occur at an average rate λ and SCOV of inter-arrival times C_a^2. Each arrival brings a random amount of work which is IID with mean $1/\mu$ and SCOV C_s^2. Let f be the amount of work that can be processed by the server per unit time. Assuming that $\lambda < \mu f$ and using the fact that the inter-arrival times and service times are much smaller than the one-hour interval, we can use the above result as an approximation for the probability that the virtual delay V (at any time during the one-hour interval) is greater than δ as

$$P\{V > \delta\} \approx e^{-\frac{2\delta(1-\lambda/(\mu f))(\mu f)^2}{\lambda(C_a^2+C_s^2)}}.$$

The above result is also known as Kingman's heavy traffic approximation. Using a QoS metric that the probability of exceeding a virtual delay greater than δ must be no more than ϵ, we get the QoS criteria

$$P\{V > \delta\} \leq \epsilon.$$

For an arbitrary server $j \in S$, we select $f_{\min} = \min_{z\in Z_j} \gamma_{jz}$ as the slowest processing rate when powered on, and $\mu_{\min} = \min_{i\in A} \mu_i$ the largest average job size. For all $i \in A$, let $C_{a,i}^2$ be the SCOV of inter-arrival time and $C_{s,i}^2$ be the SCOV of the amount of work each request brings for application i. Since requests are split among servers running application i, the effective inter-arrival time SCOV for server j which is the superposition of applications hosted on it would be at most as large as $C_{a,\max}^2 = \max\{1, \max_{i\in A} C_{a,i}^2\}$. Likewise the SCOV of the effective amount of work due to all applications on server j can be upper-bounded by $C_{s,\max}^2$ which can be obtained by solving the following mathematical program (with decision variables p_i and $X_i \; \forall \; i \in A$ and auxiliary variable

$\overline{\mu}$):

$$\begin{aligned} C^2_{s,\max} = \text{Max} \quad & \left\{-1 + \textstyle\sum_{i\in A} p_i X_i (C^2_{s,i}+1)\overline{\mu}^2/\mu_i^2\right\} \\ \text{s.t.} \quad & \textstyle\sum_{i\in A} p_i X_i/\mu_i = 1/\overline{\mu} \\ & \textstyle\sum_{i\in A} p_i X_i = 1 \\ & \textstyle\sum_{i\in A} X_i \le C_j \\ & 0 \le p_i \le 1, X_i \in \{0,1\}, \quad \forall i \in A. \end{aligned}$$

The above is based on the fact that if a fraction p_i of application i arrives at a server, then the effective SCOV of the aggregated service times is $\left\{-1 + \sum_{i\in A} p_i X_i (C^2_{s,i}+1)\overline{\mu}^2/\mu_i^2\right\}$ with $1/\overline{\mu}$ the aggregate mean workload.

Let $\rho = \lambda/(f\mu)$ then for any server j, the QoS criteria $P\{V > \delta\} \le \epsilon$ would be satisfied if we select a ρ that satisfies

$$e^{-\frac{2\delta(1-\rho)\mu_{\min} f_{\min}}{\rho(C^2_{a,\max}+C^2_{s,\max})}} = \epsilon$$

as the left hand side of the above expression is larger than $P\{V > \delta\}$. Thus

$$\rho = \left[1 + \frac{C^2_{a,\max} + C^2_{s,\max}}{2\delta\mu_{\min} f_{\min}} \log_e(1/\epsilon)\right]^{-1}.$$

For example, when $f_{\min} = 512$ kbps, $1/\mu_{\min} = 15$ kb, $\delta = 2$ seconds, $C^2_{a,\max} = 1$, $C^2_{a,\max} = 2$ and $\epsilon = 1.15 \times 10^{-5}$, we have $\rho = 0.8$ as the desired value of $\lambda/(\mu f)$. In fact for any $\rho < 0.8$, the QoS will be met. Therefore at server j, if the aggregate workload arrival rate λ/μ is lesser than ρ times the workload processing rate f, the QoS criteria will be satisfied. Although this ρ is truly for server j, we could remove the subscript by selecting the smallest ρ_j among all j or just use ρ_j instead of ρ in constraint (1d).

This brings us to Constraint (1d) at server $j \in S$ for any time interval $t \in T$. The aggregate workload arrival rate $\lambda/\mu = \sum_{i\in A} \frac{\lambda_{it}}{\mu_i} v_{ijt}$ since each application $i \in A$ arrives at rate $\lambda_{it} v_{ijt}$ during interval t bringing an average load $1/\mu_i$. Likewise the workload processing rate f at server j in interval t is $\sum_{z\in Z_j} u_{jzt}\gamma_{jz}$. Therefore by satisfying Constraint (1d), we can ensure that the QoS criterion $P\{V > \delta\} \le \epsilon$ would be met under a stochastic environment. In terms of time-variability within an interval, only the arrival rate is anticipated to vary in an interval, while the other parameters would remain constant. If the arrival rate λ_{it} varies over the interval t, then either using the largest among the varying rates or considering smaller intervals t during which λ_{it} is more or less a constant, it is possible to guarantee QoS.

Refer back to Figure 1. Some of the pre-processing described there includes all the analysis in this sub-section to obtain a suitable ρ. In a similar fashion, based on the infrastructure of the data center and all available historical information, the parameters in Table 2 can be obtained. Then based on the MIP solution we can obtain the decisions described in Table 3. The analysis itself could be somewhat conservative, and so some post-processing can be done to further improve the energy consumption or QoS or both. We envision the analysis to be performed a few times with several what-if questions answered to eventually converge to a suitable operating condition. That said, the remainder of this paper will focus on the processing step described in Figure 1 that converts the model inputs in Table 2 into model outputs or decisions described in Table 3.

5 Solution Method

As we saw in the previous section, due to the size of the problem we do anticipate solvers taking a large amount of time to obtain the solutions. In this section we propose a solution approach for solving the large scale MIP. In particular, we derive a heuristic method that provides good solutions for large instances of the problem as well as speed up the solution time. The key idea behind the heuristic method is to choose a "representative single time period" (RSTP) $\tau \in T$ from given workload traces and solve a reduced size MIP corresponding to this time period. An RSTP is determined by considering the workload traces of all the applications at each time period $t \in T$ and computing the total applications workload measure of interest, for example, the *minimum*, *average* or *maximum* workload. The reduced size or RSTP MIP based on τ is stated as follows:

$$\text{MIP}\tau : \text{Min} \sum_{j \in S} \sum_{z \in Z_j} \beta_{jz\tau} u_{jz\tau} \tag{2a}$$

$$\text{s.t.} \sum_{z \in Z_j} u_{jz\tau} = 1 \quad \forall j \in S, \tag{2b}$$

$$\sum_{j \in S} v_{ij\tau} = 1 \quad \forall i \in A, \tag{2c}$$

$$\rho \sum_{z \in Z_j} u_{jz\tau} \gamma_{jz} \geq \sum_{i \in A} \frac{\lambda_{i\tau}}{\mu_i} v_{ij\tau} \quad \forall j \in S, \tag{2d}$$

$$\sum_{i \in A} x_{ij} \leq C_j \quad \forall j \in S \tag{2e}$$

$$v_{ij\tau} \leq x_{ij} \quad \forall i \in A, \forall j \in S, \tag{2f}$$

$$\sum_{i \in A} v_{ij\tau} \leq C_j (1 - u_{j0\tau}) \quad \forall j \in S, \tag{2g}$$

$$v_{ij\tau} \geq 0, x_{ij} \in \{0,1\}, u_{jz\tau} \in \{0,1\} \quad \forall i \in A, \forall j \in S, \forall z \in Z_j. \tag{2h}$$

Let the application-server assignment solution to problem MIPτ be denoted by $\hat{x}_{ij}, \forall i \in A, \forall j \in S$. Our desire is to obtain a $[\hat{x}_{ij}]$ among alternative optimal solutions to MIPτ so that $\sum_i \sum_j \hat{x}_{ij}$ is minimized. Some commercial solvers such as CPLEX automatically present such a solution due to the way they perform branch and bound. However, one could add $\sum_i \sum_j \hat{x}_{ij}$ to the objective function realizing that $\beta_{izt} > 1 \; \forall \; j, z > 0, t$ (if not, it can be accomplished by scaling appropriately). Now, let $\mathcal{L}_0$ and $\mathcal{L}_1$ be index sets for all the application-server pairs $(i,j) \in A \times S$ such that $(i,j) \in \mathcal{L}_0$ for $\hat{x}_{ij} = 0$ and $(i,j) \in \mathcal{L}_1$ for $\hat{x}_{ij} = 1$. Since the $\hat{x}_{ij}$'s are based on the RSTP τ, we need a solution that is feasible for all time periods. To get such a solution, we substitute the representative time period solution $\hat{x}_{ij}, \forall i \in A, \forall j \in S$ into constraints (1e) and (1f) of the original problem MIP1 and only fix the assignments for all $(i,j) \in \mathcal{L}_1$ but not for all $(i,j) \in \mathcal{L}_0$. The idea is to now solve the original problem with application-server assignments based on τ, but with the flexibility of allowing the model to determine additional application-server assignments if needed, and the server frequencies (u_{jzt}), and the fraction of arrivals of each application at each server (v_{jzt}) at each time period $t \in T$. The original problem with the RSTP application-server assignments fixed for all $(i,j) \in \mathcal{L}_1$ takes the following form:

$$\text{MIP2} : \text{Min} \sum_{j \in S} \sum_{z \in Z_j} \sum_{t \in T} \beta_{jzt} u_{jzt} \tag{3a}$$

$$\text{s.t.} \sum_{z \in Z_j} u_{jzt} = 1 \quad \forall j \in S, \forall t \in T \tag{3b}$$

$$\sum_{j \in S} v_{ijt} = 1 \quad \forall i \in A, \forall t \in T \tag{3c}$$

$$\rho \sum_{z \in Z_j} u_{jzt} \gamma_{jz} \geq \sum_{i \in A} \frac{\lambda_{it}}{\mu_i} v_{ijt} \quad \forall j \in S, \forall t \in T \tag{3d}$$

$$\sum_{(i,j) \in \mathcal{L}_0} x_{ij} \leq C_j - \sum_{(i,j) \in \mathcal{L}_1} \hat{x}_{ij} \quad \forall j \in S \tag{3e}$$

$$v_{ijt} \leq x_{ij} \quad \forall (i,j) \in \mathcal{L}_0, \forall t \in T \tag{3f}$$

$$v_{ijt} \leq \hat{x}_{ij} \quad \forall (i,j) \in \mathcal{L}_1, \forall t \in T \tag{3g}$$

$$\sum_{i \in A} v_{ijt} \leq C_j (1 - u_{j0t}) \quad \forall j \in S, \forall t \in T \tag{3h}$$

$$v_{ijt} \geq 0, u_{jzt} \in \{0,1\} \quad \forall i \in A, \forall j \in S, \forall z \in Z_j,$$

$$\forall t \in T, x_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{L}_0 \tag{3i}$$

Observe that constraints (3e) and (3g) explicitly enforce the requirement that $x_{ij} = 1, \forall (i,j) \in \mathcal{L}_1$. An alternative way to achieve this requirement is to simply append it as a constraint to the original problem MIP1. However, by explicitly representing this requirement as in problem MIP2,

we reduce the number of decision variables in the original problem by $|\mathcal{L}_1|$. Now by solving the relatively smaller size problem MIP2, we can obtain our heuristic solution. We should note that problem MIP2 can be infeasible depending on the RSTP τ used. In that case a different RSTP has to be used until a feasible solution can be found. Based on our computational results, the solutions provided by MIPτ for all the trace data sets we considered are such that the number of applications assigned to some servers is less than C_j. Also, the solutions for u_{jz}'s tend to choose middle frequencies (around the 5th among nine alternatives) most of the time.

Let us now define the parameters needed in determining the RSTP τ. Recall that the main input data for each application i is a workload trace for all $t \in T$. Therefore, we propose using the overall *maximum* workload across all time periods. The aim is to find a time period τ that would provide a feasible, and preferably, close to optimal assignment of applications to servers (x_{ij}'s). Thus, in general, we can provide a feasible application-server assignment by using the overall maximum workload across time to determine τ. Further, determining τ should be computationally easy to do. Let ω_{it} denote the average workload per second brought by application i, mathematically written as $\omega_{it} = \frac{\lambda_{it}}{\mu_i}$. Then the overall workload at time $t \in T$, denoted ω_t, is given by $\omega_t = \sum_{i \in A} \omega_{it}$. Therefore, the RSTP corresponding to the overall *maximum* workload across time, τ_{max}, is

$$\tau_{max} = \underset{t \in T}{\text{argmax}} \ \omega_t. \tag{4}$$

Instead of using the maximum overall workload to determine a RSTP, one can consider using the overall *minimum* or *average* workload, or the total workload corresponding to a set of applications that contribute a certain percentage (e.g. 80%) to the total workload. Based on our experiments, the maximum workload provided the best performance as expected. Next, we give a formal outline of our heuristic. Since this method finds a non-zero assignment of applications to servers, we refer to it as the 'Single-Period Heuristic for Initial Non-zero X' (SPHINX) algorithm.

Basic SPHINX Algorithm:

Step 0. Initialization. Set $U \leftarrow \infty$.

Step 1. Determine RSTP τ. Use *maximum* workload to get $\tau \leftarrow \tau_{max}$ via Equation (4).

Step 2. Form and Solve MIPτ. Use RSTP τ and solve problem MIPτ (2). Let $\hat{x}_{ij}, \forall i \in A, \forall j \in S$ be the optimal application-server assignment solution. Create the sets $\mathcal{L}_0$ and $\mathcal{L}_1$.

Step 3. Form and Solve MIP2. Using the solution $\hat{x}_{ij}, \forall i \in A, \forall j \in S$ from Step 2 and the sets $\mathcal{L}_0$ and $\mathcal{L}_1$, form and solve problem (3). If problem (3) is *feasible*, let z denote the optimal objective function value and (x, v, u) be the optimal solution vector for the application-server assignments, fraction of arrivals of an application assigned to a given server, and server frequencies, respectively. Go to step 4. If the problem is *infeasible* set $z = \infty$ and go to step 5.

Step 4. Update Incumbent Solution. Set $U \leftarrow \min\{z,\ U\}$. If U is updated set the incumbent solution $(x^*, v^*, u^*) \leftarrow (x, v, u)$.

Step 5. Termination. If $U < \infty$ stop and report (x^*, v^*, u^*) as the solution. Otherwise, if $U = \infty$, no solution could be found using the *maximum* workload RSTP, go to step 2 and try a different workload RSTP (e.g. minimum or average).

The intuition behind our heuristic approach is to find a single time period where the aggregated workload of all the application traces yields application-server assignments that are feasible for *all* time periods and allow for varying the server frequencies towards satisfying workload at other times. That is why we refer to such a single time period to as RSTP. Furthermore, if the single time period application-server assignments cannot satisfy workload demand at any other time, then we want to perform additional application-server assignments to meet the workload demand. This is possible if the RSTP application-server assignments are such that, 1) there is extra capacity on the servers, i.e., $\sum_{i \in A} x_{ij} \leq C_j$ to allow for additional applications to be assigned to some server, and/or 2) there is enough servers not assigned any applications that can now be assigned applications to satisfy the workload demand. Thus the SPHINX algorithm is expected to perform well in cases where the data center has enough server capacity to meet the highest workload demand that can be received by the data center. In fact this is usually the case in practice. The SPHINX algorithm may not be expected to perform well when there is not sufficient server capacity and the RSTP application-server assignments are such that all servers are assigned applications up to C_j and have to run at the highest frequency. In that case one can consider solving MIP1 directly if possible.

The basic SPHINX algorithm can be extended to allow for using different RSTPs until a feasible solution can be found. We implemented the SPHINX algorithm and experimented with several RSTP types. The overall maximum workload based RSTP provided the best performance followed by the average and minimum, in that order. To provide a benchmark for the SPHINX algorithm to solve the integrated problem, we considered the traditional approach of separately making application-server assignments (virtualization) (Petrucci et al. [25], Bertini and Leite [1]),

dynamically assigning fraction of application arrivals to servers over time (routing), and assigning server frequencies at each time period (DVFS and on/off). We implemented this approach using a two-phase heuristic described next.

Two-Phase Heuristic:

Phase One:

Step 1. Sort the applications in nonincreasing order of total workload (i.e. for any $i, k \in A$, application i has a higher workload than k if $\sum_{t\in T} w_{it} > \sum_{t\in T} w_{kt}$).

Step 2. Assuming all servers will run at the highest frequency at all times, determine the minimum number of servers n_i each of the sorted applications i must be assigned (the number of servers would also be non-increasing across the sorted applications) to satisfy the worst case workload, i.e. $\max_{t\in T} w_{it}$ for any application $i \in A$.

Step 3. Choose applications one by one using the sorted list in step 1 and place application $i \in A$ in $\min(2, n_i)$ servers while ensuring that no more than $(C_j - 1)$ applications are assigned to each server $j \in S$.

Step 4. Following the order in the sorted applications list in step 1, if $n_i > 2$ then arbitrarily assign application i to $n_i - 2$ servers while making sure that (a) application $i \in A$ is not already assigned to server $j \in S$ in step 3, and (b) capacity C_j is not exceeded. Note that if $n_i \leq 2$, sufficient copies of application $i \in A$ are already assigned to servers in step 3. Set $x_{ij} = 1$ if application i is assigned to server j and 0 otherwise.

Step 5. Let $z_j^* = \arg\max_z \gamma_{jz}$ for all j, i.e. the index corresponding to maximum speed. Initially set $u_{jz_j^*t} = 1$ and $u_{jzt} = 0$ when $z \neq z_j^*$ for each server j for all $t \in T$. If $x_{ij} = 0$ then set $v_{ijt} = 0 \; \forall \; t \in T$. Otherwise set $v_{ijt} = x_{ij}(1 - u_{j0t})/(\sum_{k\in S} x_{ik}(1 - u_{k0t}))$ for all $t \in T$ so that the load from application i is balanced across all powered on servers hosting it.

Phase Two:

Step 1. For each server $j \in S$ at each time $t \in T$, compute total workload at each period of time based on v_{ijt} values from Phase One, i.e. $\sum_{i\in I} \frac{\lambda_{it}}{\mu_i} v_{ijt}$.

Step 2. For each server j, compute the minimum server capacity γ_{jz} required to satisfy the constraint $\sum_{i\in I} \frac{\lambda_{it}}{\mu_i} v_{ijt} \leq \rho\gamma_{jz} \quad \forall t \in T$. Set the corresponding $u_{jzt} = 1$ and all other u_{jzt} to zero.

Step 3. For each $t \in T$, power off servers one by one (starting with the server with the smallest frequency). Obtain new v_{ijt} using step 5 of Phase One and new u_{jzt} via step 2 of Phase Two.

Step 4. Compute the objective value based on the updated u_{jzt} values.

Next we report on our computational study and discuss some insights into our findings regarding the advantages of using the integrated approach versus the non-integrated approach, i.e. the two-phase heuristic.

6 Computational Results

In this section we report on several numerical experiments to: (a) test a direct solution method based on synthetic traces and compare the performance with the SPHINX algorithm; (b) solve instances with real traces using the SPHINX algorithm; and (c) answer the research questions raised in Section 3.

6.1 Design of Experiments

Recall the inputs to the MIP are tabulated in Table 2. We designed a set of experiments by systematically varying the inputs. However, since the number of input variables is prohibitively large, we set specific values for some of the parameters which we felt would not affect the insights. In particular, for all our experiments: $|A|/|S| = 2$ meaning there are twice as many applications as servers; $|T| = 24$ corresponding to the 24 hours of a planning horizon of 1 day; there are eight possible frequencies a server can run at, then including the off state of zero frequency we have $|Z_j| = 9$ for all $j \in S$; not more that 4 applications can be assigned per server, i.e. $C_j = 4$ for all $j \in S$; the target load ρ for the data center is assumed to be 0.8 which corresponds to a medium traffic intensity from a queueing standpoint.

Although the model allows for a lot of flexibility in terms of cost, including the ability to set different prices at different times of the day, for the purposes of the experimental design we assumed that β_{jzt} for a given z stays a constant across servers and across time. We used a cubic order relation (based on Chen et al. [5]) for β_{jzt} of the form $\beta_{jzt} = k_0 + k_1 z^3$, if $z > 0$ and $\beta_{j0t} = 0$ for all $j \in S$ and $t \in T$. The parameters k_0 and k_1 are constants resulting in

$$[\beta_{j0t} \;\; \beta_{j1t} \;\; \ldots \;\; \beta_{j8t}] = [0.0 \; 60.0 \; 63.0 \; 66.8 \; 71.3 \; 76.8 \; 83.2 \; 90.7 \; 100.0].$$

Finally, $\omega_{it} = \frac{\lambda_{it}}{\mu_i}$ denotes the average workload per second brought by application i.

Thus there are only three sets of parameters that would be varied in the experimental design $|A|$, γ_{jz} for all $j \in S$ and $z \in Z_j$, as well as ω_{it}. We select γ_{jz} in our experimental design so that we maintain the ratio

$$[\gamma_{j0t} : \gamma_{j1t} : \ \ldots \ : \gamma_{j8t}] = [0 : 519.94 : 583.07 : 646.21 : 709.34 : 772.48 : 835.62 : 898.75 : 965.60]$$

for all $j \in S$. Also, $|A|$ values we considered were 20, 30, 40 and 50 to illustrate various sizes of data centers from small to medium-large. Thus our other parameter ω_{it} ends up being the main driver of the experimental design.

That said, it is critical to notice that there are a large number of ω_{it} values to perform a thorough design of experiments study. Hence we only considered three summaries. For each application we considered the variability of ω_{it} across time using the following metric $C_i^2 = \frac{\sum_{t \in T}(\omega_{it} - \overline{\omega}_i)^2}{(|T|-1)(\overline{\omega}_i)^2}$, where $\overline{\omega}_i = \frac{1}{|T|}\sum_{t \in T} \omega_{it}$. Likewise for every time slot we considered the variability of ω_{it} across applications $i \in A$ using the following metric $C_t^2 = \frac{\sum_{i \in A}(\omega_{it} - \overline{\omega}^t)^2}{(|A|-1)(\overline{\omega}^t)^2}$, where $\overline{\omega}^t = \frac{1}{|A|}\sum_{i \in A} \omega_{it}$. Finally we considered the correlation between application i and k for any $i \in A$ and $k \in A$ such that $k \neq i$ using

$$C_r(i,k) = \frac{\frac{1}{|T|}\sum_{t \in T}(\omega_{it}\omega_{kt}) - \overline{\omega}_i\overline{\omega}_k}{\sqrt{\sum_{t \in T}(\omega_{it} - \overline{\omega}_i)^2}\sqrt{\sum_{t \in T}(\omega_{kt} - \overline{\omega}_k)^2}}. \tag{5}$$

Notice that C_i^2 and C_t^2 are terms similar to squared coefficient of variation, and $C_r(i,k)$ is similar to coefficient of correlation between i and k. However, it is crucial to realize that ω_{it} is not a random variable. It is in fact a deterministic quantity denoting the expected value of the workload per unit time from application i arriving in time interval t. It is obtained by forecasting based on historical data. For our design of experiments we consider high and low variability across applications as well as time. In particular if C_i^2 or C_t^2 is around 4 or higher, then we say that the variability is high across time of applications respectively. Likewise if C_i^2 or C_t^2 is around 0.5 or lower, then we say that the variability is low across time of applications respectively. In a similar fashion if $C_r(i,k)$ is over 0.8, between 0.4 and -0.4, and below -0.8 we say that applications i and k are positively correlated, independent (or weakly correlated) and negatively correlated respectively.

Because of very limited access to real data (presented in Section 6.4 and Appendix) we synthetically created workload data that mimic real data in overall form. In our experiments we consider the whole spectrum of feasible factorial design of experiments with respect to variability (high

and low across time) and correlations (positive, independent and negative-across-groups). In the negative across groups case, applications are divided into two positively correlated groups, however any trace from one group would be negatively correlated with any trace from the other group.

Table 4 summarizes the entire set of test instances we created. We considered *two* sets of experiments each of those including *two* sets of instances. In the first set of experiments, the number of applications ($|A|$), gamma values (γ_{jz}), and replications were increased while keeping the average workload (ω_{it}) characteristics constant (test instances 1,2, and 3 in Table 4). Characteristics of average workload for this set of experiments were chosen to be similar to those present in real traces, i.e., high variability across applications and low variability across time (HL) with applications that were weakly correlated (independent). In the second set of experiments, we considered the whole spectrum of feasible factorial design of experiments especially with respect to variability and correlations across applications and time (test instance 4 in Table 4). In the table, LL stands for low variability across applications and low variability across time. Similarly, LH stands for stands for low variability across applications and high variability across time, and HH stands for stands for high variability across applications and high variability across time. The number of applications, replications, and gamma values were kept fixed, i.e., $|A| = 20$, single replications, and minimum γ_{jz} for feasible MIP instances.

Table 4: Summary of test instances based on simulated traces

Test Set	**No. of Applications**	**No. of Servers**	**Variability & Correlation**
1	20 30 40 50	10 15 20 25	HL, Independent
2	20 30 40 50	10 15 20 25	HL, Independent
3	20 30 40 50	10 15 20 25	HL, Independent
4	20	10	LH, Positive LL, Both Positive & Negative LL, Independent LH, Positive HH, Positive HL, Independent HL, Positive HL, Both Positive & Negative

L: Low, H: High

We implemented the SPHINX algorithm in C++ using CPLEX 11.0 Callable library and tested

it on several instances. In addition to implementing the RSTP based on the maximum workload, we also implemented one based the average workload for comparison. We also compared the SPHINX implementations against the CPLEX MIP solver applied to the same set of instances for comparison. Experiments were run on a Dell Optiplex 755 computer with 2 Intel (R) Core (TM) 2 Quad CPU Q 9650 processors at 3.00 GHz each with 8.0 GB of RAM. Next we report on our findings.

6.2 Results for Integrated Versus Non-Integrated Approach

The first set of experiments was aimed to address the issue of whether the integrated approach provides better solutions than the traditional non-integrated approach. Therefore, we conducted experiments based on simulated synthetic data to address this question. In particular, we considered a variety of experiments to understand the conditions when the non-integrated approach would perform reasonably well. However, we only present a limited set of results due to space restrictions. In fact the results are for cases when the non-integrated approach is expected to perform well. They are when the application load is correlated with each other (either both positive and negatively correlated pairs or just positively correlated ones), and when the variability in load across applications is low (else clustering is unlikely). We consider both low and high variability across time for all applications.

Notice from Tables 5 and 6 that the objective value for our integrated approach is significantly better than the non-integrated approach (two-phase heuristic). It is crucial to realize that the objective of 24,000 for *Phase 1* of the two-phase heuristic is the resulting cost due to running all servers at the maximum speed. Of course the CPU time of the two-phase heuristic is quite negligible. In summary, the two-phase heuristic is reasonable to get a first cut solution but the quality of the solution can be significantly improved using the integrated approach. Also notice from the tables that the SPHINX algorithm performs almost as good as the CPLEX solution in a much smaller amount of time. Thus we conclude that an integrated approach is worth pursuing and perform extensive testing next to understand what causes this.

Table 5: Integrated versus non-integrated approach for LL-Both

	Objective	**CPLEX % Gap**	**% Opt. Gap**	**CPU Time (sec.)**
CPLEX	18,088	1.33	0.00	25,200.00
SPHINX	18,095	1.37	0.04	3,341.99
Two-Phase Heuristic:				
Phase 1	24,000	0.00	32.69	0.19
Phase 2	19,759	0.00	9.24	0.03

Table 6: Integrated versus non-integrated approach for LH-Positive

	Objective	**CPLEX % Gap**	**% Opt. Gap**	**CPU Time (sec.)**
CPLEX	8,926	0.03	0.00	25,200.00
SPHINX	8,933	0.07	0.08	3,600.40
Two-Phase Heuristic:				
Phase 1	24,000	0.00	168.87	0.03
Phase 2	12,851	0.00	43.97	0.02

Finding 1 *The integrated approach, albeit much slower that the non-integrated approach, results in a significant gain (up to 40%) in objective function value (i.e. cost per unit time) compared to the non-integrated approach.*

6.3 Comparing Representative Periods: Maximum and Average

To further test the performance of the SPHINX algorithm we applied it to test sets 1 and 4 in Table 4. Figure 2 shows a plot of objective function values obtained by the direct CPLEX MIP solver and SPHINX using RSTP based on average workload (average-RSTP) and on maximum workload (maximum-RSTP). As can be seen in the plot, SPHINX maximum-RSTP provides the best results across all the test instances. CPLEX could not get a feasible solution for the largest test instance 50 APPS (50 applications) within the time limit (25,000 seconds) while the SPHINX algorithms obtained the same objective value. For test instance HL, Independent, both CPLEX and SPHINX maximum-RSTP obtained the same solution. However, SPHINX average-RSTP could not solve this instance. Taking a closer look, for this particular instance the time period with the average workload is not representative of the workload across the horizon. Thus the application-server assignment solution based on the representative time period with average workload is not feasible for the overall problem. The corresponding CPU times for the test sets 1 and 4 are plotted in Figure 3. The CPU time is not reported for instances that could not be solved by a given algorithm. The average CPU time for SPHINX maximum-RSTP is 5,322 seconds, compared to 6,216 seconds for SPHINX average-RSTP. CPLEX was run to the time limit of 25,000 seconds as it could not obtain a solution by the time both the other algorithms had terminated.

We also tested the performance of SPHINX Maximum-RSTP versus the direct solver CPLEX on instances of the same size but different parameters. A CPU time limit of 10,800 seconds was imposed. We made ten replications of instances corresponding to test set 3 in Table 4 with 30 applications. The results are reported in Table 7. In the table we report the objective value, MIP gap and % gap relative to CPLEX. As can be seen in the table, SPHINX maximum-RSTP obtains lower objective values than CPLEX. In particular, CPLEX could not find a feasible solution for

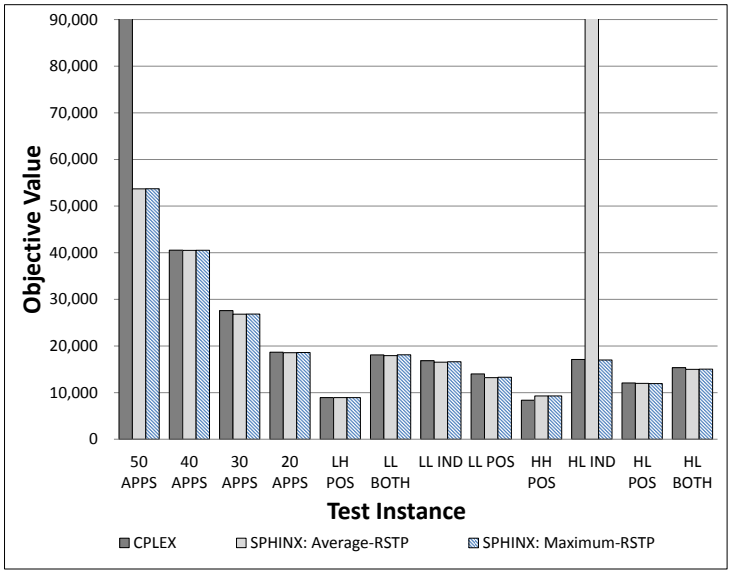

Figure 2: Objective values for instances solved using SPHINX and CPLEX

replications 4, 7, and 8. However, SPHINX is able to solve all the replications to within relatively small MIP gaps. These results are consistent with those reported in Figure 2 in which SPHINX maximum-RSTP performs consistently better than the direct solver CPLEX for different size test instances. We obtained similar results even for test instances with 40 and 50 applications with high-low variability and independent applications.

Table 7: Instance with 30 applications and high-low independent traces with 10 replications

	CPLEX MIP Solver			**SPHINX Maximum-RSTP**			
Rep	Objective Value	MIP Gap	Solution Time	Objective Value	MIP Gap	% Gap	Solution Time
1	23,586	13.31%	10,800	21,570	3.60	-8.55%	10,800
2	23,297	10.81%	10,800	21,889	3.77	-6.05%	10,800
3	24,388	7.27%	10,800	23,196	2.24	-4.89%	10,800
4	∞	-	10,800	25,753	0.34	-	10,800
5	29,816	2.83%	10,800	29,183	0.54	-2.12%	10,800
6	23,138	19.24%	10,800	19,908	4.33	-13.96%	10,800
7	∞	-	10,800	19,882	4.53	-	10,800
8	∞	-	10,800	26,088	0.67	-	10,800
9	23,486	12.28%	10,800	21,901	5.20	-6.75%	10,800
10	26,599	4.64%	10,800	25,868	1.15	-2.75%	10,800

Finding 2 *The time period with the maximum aggregate workload is an ideal choice for representative time period as it performs the best among any other choice of time period when it comes to minimizing the energy cost per unit time while satisfying the constraints on resources and capacities.*

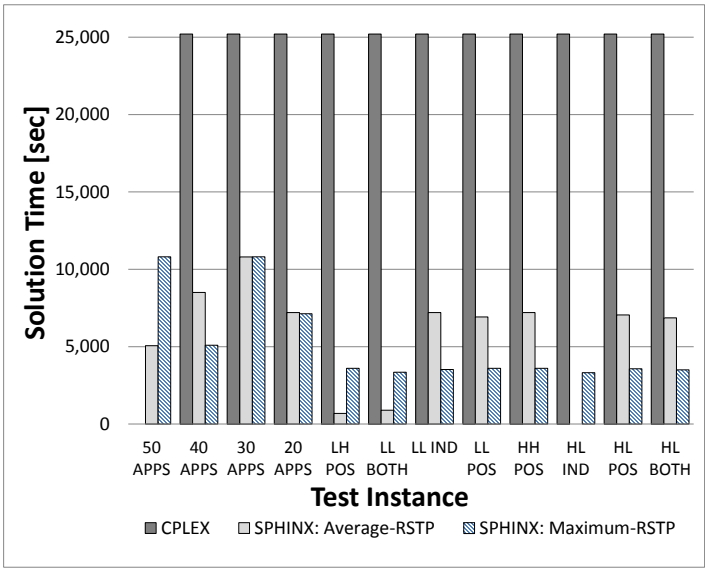

Figure 3: Solution time for instances solved using SPHINX

In addition, when the problem is feasible, SPHINX with maximum time period always outperforms solving the entire problem as an MIP with a reasonable time constraint.

6.4 Detailed Discussion of Results Using Real Traces

We now report the findings from our analysis using real traces from real web-server applications that are publicly available. This set of real traces consists of 20 applications which we seek to optimally allocate to 10 servers (see Appendix). In addition to the optimal allocation of application to servers, we seek the optimal frequency that each server should run at for each time period as well as the optimal routing strategy. That is, we seek to determine which servers are called upon to satisfy requests for each application for each time period. By doing so, we can gain insight about our problem and answer the research questions posed in Section 3. We use $|A| = 20$ applications and $\gamma_{j8} = 100$. From the tables in the Appendix we can see that for most applications C_t^2 is higher than 4 and C_a^2 is lower than 0.5, hence we say that the variability is high across applications and low across time. Some applications were positive correlated, and some pairs were negatively correlated but most applications were weakly correlated (independent). Using our MIP, the optimal allocation of applications to servers for this set of real traces can be obtained, which is presented in Table 8.

Notice, in Table 8, that each server is indexed by j while each application is indexed by i. Therefore, each element of the table tells us if an application is assigned to a particular server. For example, we can see that application 3 is assigned to server 4 since the corresponding element for

Table 8: Allocation of applications to servers

	j									
i	1	2	3	4	5	6	7	8	9	10
1	1	1	1	0	0	0	1	1	1	1
2	0	1	0	0	1	0	0	0	0	0
3	0	0	0	1	1	0	0	0	0	1
4	0	0	1	0	0	1	0	0	0	0
5	0	0	1	1	0	0	0	1	0	0
6	1	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	0	1	0	0
8	0	0	0	0	0	0	1	1	0	0
9	0	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	1
11	1	0	0	0	0	0	1	0	0	0
12	0	1	0	0	0	0	0	0	0	0
13	0	0	0	1	0	0	0	0	0	0
14	0	0	1	0	0	1	0	0	0	0
15	0	0	0	1	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	1	0
17	0	0	0	0	0	0	0	0	1	0
18	0	0	0	0	0	1	0	0	0	1
19	0	0	0	0	0	0	0	0	1	0
20	1	0	0	0	0	0	1	0	0	0

$i = 3$ and $j = 4$ is equal to 1. Similarly, we can see that application 3 is not assigned to server 1 since the corresponding element for $i = 3$ and $j = 1$ is equal to 0. Notice that each element of Table 8 corresponds to values for the decision variable x_{ij}. From Table 8 we can see that each server has 4 applications assigned to it. This is because the parameter C_j was set to 4 while solving this problem, which limits the number of applications that can be assigned to each server to a value of 4. Taking a closer look at servers 1, 3, 6, and 9, we can see which applications are assigned to these servers. Applications 1, 6, 11, and 20 are assigned to server 1, while applications 1, 4, 5, and 14 are assigned to server 3. Additionally, applications 4, 10, 14, and 18 are assigned to server 6, while applications 1, 16, 17, and 19 are assigned to server 9.

Finding 3 *There are no clusters of applications formed that are replicated in many servers.*

We now turn our attention to understanding how the frequency at which each server runs at changes over time. We select a sample from the set of servers to examine how the frequency changes over time, which can be seen in Figure 4. Figure 4 shows the frequency over time for servers 1, 3, 6 and 9. By looking at Figure 4, we can see that server 1 is off at time periods 5, 6, 7, 9 and 24 i.e. when frequency index is 0. We can also see that server 1 runs at frequency 5 for a majority of time. Note that this does not mean that the frequency most often used is 5, but that it means that the most often used frequency is the 5th frequency from the set of available frequencies. Similarly, we can also see how the frequency of server 3 changes over time. We can see that server 3's frequency

Figure 4: Frequency of servers 1, 3, 6 and 9 over time

ranges from being off (during time periods 1, 4, 6, 7, 10, 11 and 24) to the maximum frequency index of 8. We also see similar behavior for server 6. From Figure 4 we can see that server 9 behaves in a manner similar to that of server 1. That is, that the majority of the time, server 9 is run at frequency 5.

Finding 4 *Frequency index 5 is the mode. Also, powering on/off and DVFS both occur, and all frequencies (not just the maximum and minimum) are used.*

Now, we discuss the optimal splitting of application requests to servers. Again, by choosing a sample from the set of applications, we discuss how requests for applications are routed to a particular set of servers. From Figure 5, we see how requests for application 1 are distributed to servers 1, 2 and 7. From Figure 5, we can see that requests for application 1 are not routed to server 1 during time periods 5, 6, 7 and 9. This does not mean that server 1 is off during these time periods, but that requests for application 1 are satisfied by some other server that houses the application. In this example, we can see that a fraction of arrivals for application 1 are routed

Figure 5: Fraction of arrivals of Application 1 on Servers 1, 2 and 7

to servers 2 and 7 during these time periods. Similarly, for server 7, we can see that arrivals of application 1 are not routed to server 7 during time periods 3 and 23, but instead a fraction of those arrivals are routed to servers 1 and 2. For application 14, we see a different routing scheme to servers 3 and 6. The routing of application 14 to servers 3 and 6 can be seen in Figure 6.

Notice that the fraction of arrivals of application 14 routed to server 3 jumps back and forth between 0 and 1. Similarly, for server 6, the fraction of arrivals also jumps back and forth between 0 and 1. This is because application 14 is only assigned to servers 3 and 6 which can be seen by from Table 8. From Figure 6, we can see that in time period 2, all arrivals of application 14 are routed to server 3 while no arrivals are routed to server 6. This is because the fraction of arrivals of application 14 to server 3 equal 1 while the fraction of arrival of application 14 to server 6 equals 0. Notice that in time period 8, the fraction of arrivals of application 14 assigned to server 3 is equal to 0.175 while the fraction of arrivals assigned to server 6 is equal to 0.825, which can be observed in Figure 6. Since application 14 is only on servers 3 and 6, these fractions must sum to 1, which is

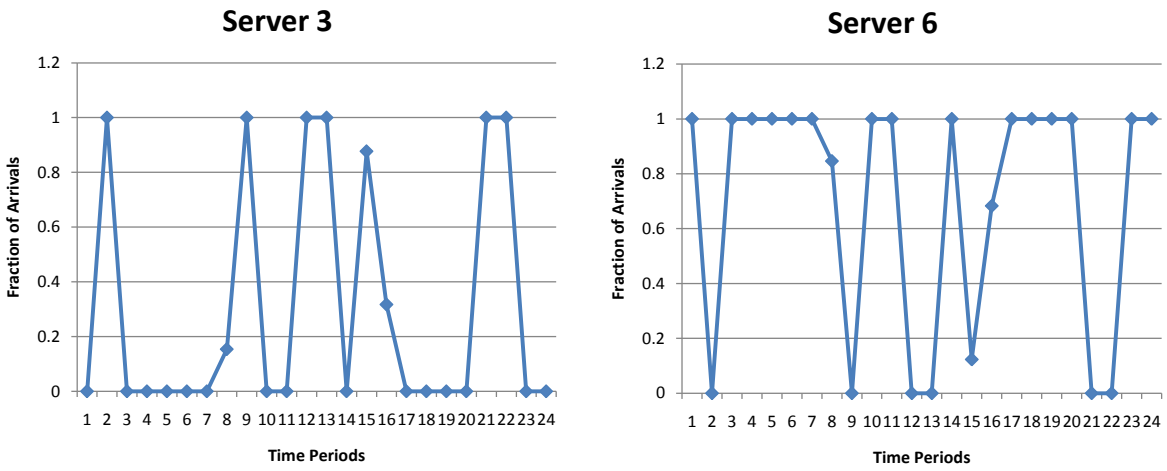

Figure 6: Fraction of arrivals of Application 14 on Servers 3 and 6

the case. We can see similar routing of application 14 in time periods 15 and 16. For the remaining time periods, we can see that application 14 is either routed to server 3 or server 6, but not both.

Finding 5 *Instead of load balancing, the loads are in fact sometimes as imbalanced as can be for an application. Applications clearly are not split across servers in a constant fashion across time.*

6.5 Answers to the Research Questions

We now address the research questions posed in Section 3. While these results and findings are illustrated for real traces, we found our conclusions to be consistent across all experiments performed for simulated traces as well.

1. *Are servers clustered across applications?* Answer: No.

   By examining Table 8 we see that no clusters appear in the assignment of applications. If clusters were present, we would see two or more columns in the table with the same values for each row. This shows that no two servers have the same set of applications assigned to them, hence, no clustering occurs. A heterogeneous mixture of application to server assignment emerges as opposed to clusters in order to capture the variability in workload across time and across applications at the same maintaining more or less balanced aggregate load at each server that is on.

2. *Are applications positively or negatively correlated assigned to the same server?* Answer: No.

   Consider applications 2 and 10 which have a positive correlation coefficient of 0.9622. By looking at Table 8 we can see that application 2 is assigned to servers 2 and 5, while application

10 is assigned to servers 6 and 10. Similarly, consider applications 3 and 14 which have a negative correlation coefficient of -0.8010. From Table 8, we can see that application 3 is assigned to servers 4, 5 and 10 while application 14 is assigned to servers 3 and 6. This shows that applications that are positively correlated are not necessarily assigned to the same set of servers. Similarly, applications that are negatively correlated are not assigned to the same set of servers. The workload brought by an application seems to be the most crucial factor in determining assignments especially when the utilization is fairly high (which is typical when unnecessary servers are removed). For example, if two correlated applications bring in a large workload they may not be on the same server due to capacity limitation, while having one of them bring in a small workload renders their pairing inconsequential.

3. *Are assignments based on peak load a good idea?* Answer: Yes.
   Using the SPHINX algorithm, we have found that in most cases making assignments based on peak load is, in fact, a good idea. Recall that in the SPHINX algorithm we consider total workload across all the applications at a given period and use the time period with maximum total workload (peak) as the representative time period (RSTP). We observe that 80% or more of the accumulative workload at the time period with the maximum workload across time (Max-RSTP) is achieved by the same subset of applications that account for a similar percentage of accumulative workload in the workload profile. Thus application-server assignments based on the RSTP are typically feasible at other time periods. Hence by judiciously performing operational decisions, the overall cost is minimized.

4. *Are servers running at the same frequency in a given time interval?* Answer: Yes.
   By looking at Figure 4 we can see which frequency servers 1, 3, 6 and 9 run at for each time interval. With the exception of server 6, other servers predominately run at frequency 5 in most time intervals, suggesting that servers do more-or-less run at the same frequency in a given time interval. Given a server j is 'on' (i.e. $z > 0$), the energy cost β_{jzt} is convex in z. Therefore, it would be optimal to run all the 'on' servers at the same frequency if one could appropriately route applications to result in an aggregated load balance across servers.

5. *Is the workload for each application split across servers in a constant fashion over time?* Answer: No.
   To achieve a balance of the aggregate load across all the 'on' servers, the workload for each application must be split differently over time to adapt to the time-varying workload across applications. For example, by examining Figure 5 we can see that the allocation of application

1 to servers 1, 2 and 7, respectively, is not constant over time. These figures show how application 1 is routed to the different servers and we can see that application 1 is not split across servers in a constant fashion. Similarly, by looking at Figure 6, we can see how application 14 is split across servers 3 and 6, respectively. In this case, we can see that there are times when all of application 14 is routed to servers 3 and 6. This is indicated in the figures where the fraction of arrivals is equal to 1. We can see that during time periods 3 through 7 that application 14 is not routed to server 3, but instead it is routed to server 6. We can see this by noting that the fraction of arrivals for application 14 to server 3 during time periods 3 to 7 is 0. Meanwhile, the fraction of arrivals of application 14 to server 6 during time periods 3 to 7 is 1.

6. *Is it sufficient to just consider powering on/off without doing DVFS?* Answer: No.
   Although powering off significantly reduces energy consumption, there are some servers that can never be powered off, especially if they contain applications that are not on any other servers. Also, when the granularity is large (such as one hour), the QoS criteria would not be met by merely powering servers on and off. Figure 4 shows how the frequency of servers 1, 3, 6, and 9 are utilized over time. From this figure, we can see that these servers are on most of the time. Meanwhile, these servers are not run at the highest frequency in each time period. In fact, the frequency each server runs at varies significantly over time. In this case, we can see that only server 3 is run at the highest frequency in two time intervals, namely, time interval 13 and 22.

7. *Is the selected frequency during an interval of bang-bang nature, i.e. the highest or lowest possible frequency?* Answer: No.
   The optimal frequency to run a server is the lowest possible one for which the capacity constraints would be satisfied. Also, the granularities are so large that we do not consider the effects of decisions in one interval to impact the next. In Figure 4 we can see that the frequency values servers 1, 3, 6, and 9 take on vary over time. For example, consider server 6; although server 6 is turned off in time intervals 2 and 13, the frequency at which this servers runs at over time never reaches the maximum frequency. In fact, the frequencies selected range over the entire range of allowable frequency values, except for the maximum frequency, and spend the majority of the time at frequencies around the middle or lower portion of the allowable frequency range. This supports the notion that a bang-bang type policy is suboptimal.

8. *Is it necessary to consider all the applications while making assignment, routing and frequency selection?* Answer: Yes.

   Even though applications that bring higher load are hosted on more servers than applications that bring in lower load, it is necessary to consider all applications while making assignments, routing and frequency selections. Not considering some applications, especially those with lower workloads, can often lead to infeasible solutions. Since the number of applications at a server, say j, can be at most C_j, by considering only the elephant applications we run the risk of not allocating the mice applications to any or at most one server (if we ensure the capacity is not reached). That would result in having to let all the servers powered on all the time and not perhaps the optimal mix of assignments.

7 Concluding Remarks

Energy consumption can be significantly reduced in data centers by appropriately handling resources in servers. Most previous studies propose the operation of data centers at separate granularities of time, i.e. by considering the strategic problem of mapping applications to servers (possibly using virtualization) separate from the operational problem of DVFS, routing and powering off servers. To the best of our knowledge, there is no study that proposes a method that combines both granularities. This paper uses a unified methodology that combines virtualization, speed scaling, routing, and powering off servers technologies under a single framework to efficiently operate data centers.

Our methodology uses mixed integer programming to decide how best to allocate applications to servers to minimize the average energy cost of operations per unit time. Further, a heuristic method is designed to reduce the computational time when solving large instances of the problem. The computational results show that assigning applications to servers based on peak load is a good idea and that results can be obtained using our methodology in reasonable times. However, we find that several other assumptions made in the literature are not valid. In particular, applications are not clustered to form meta-applications which are replicated in multiple servers, server frequencies are not bang-bang in nature, we need to consider both DVFS and powering on/off, etc.

This work opens up several future research directions. For example, there are several other systems that exhibit high degree of variability across time and attributes. To manage them in a holistic and systematic manner so that appropriate strategic decisions are made that are operations-aware would go a long way in efficiently utilizing such systems. In particular, systems where there

are conflicting goals such as minimizing energy and maximizing performance at the lowest cost calls for approaches such as considered in this paper. These results can also be seamlessly extended to sensor networks, energy-intensive manufacturing enterprises, and building management systems.

Appendix

This appendix gives a summary description of a sample of the traces we used for our experiments, which are referred to in Sections 6.3 and 6.4. The traces described in Section 6.4 are based on traces of 20 real world applications. They were collected from publicly available hourly workloads for various applications in a single day. Figure 7 shows a plot of the traces with the corresponding numerical data reported in Tables 9 and 10. Since application 1 dominates the figure, it appears as though the other applications are a constant, but that is not the case (it is only because of scaling effects). This is evident from Table 9, however, due to space restrictions, we only provide a sample of this data. Notice the C_i^2 and C_t^2 values to the right and bottom of the table to appreciate the variability. Also, Table 10 provides the values of $C_r(i,k)$ for all $i \in A$ and $k \in A$.

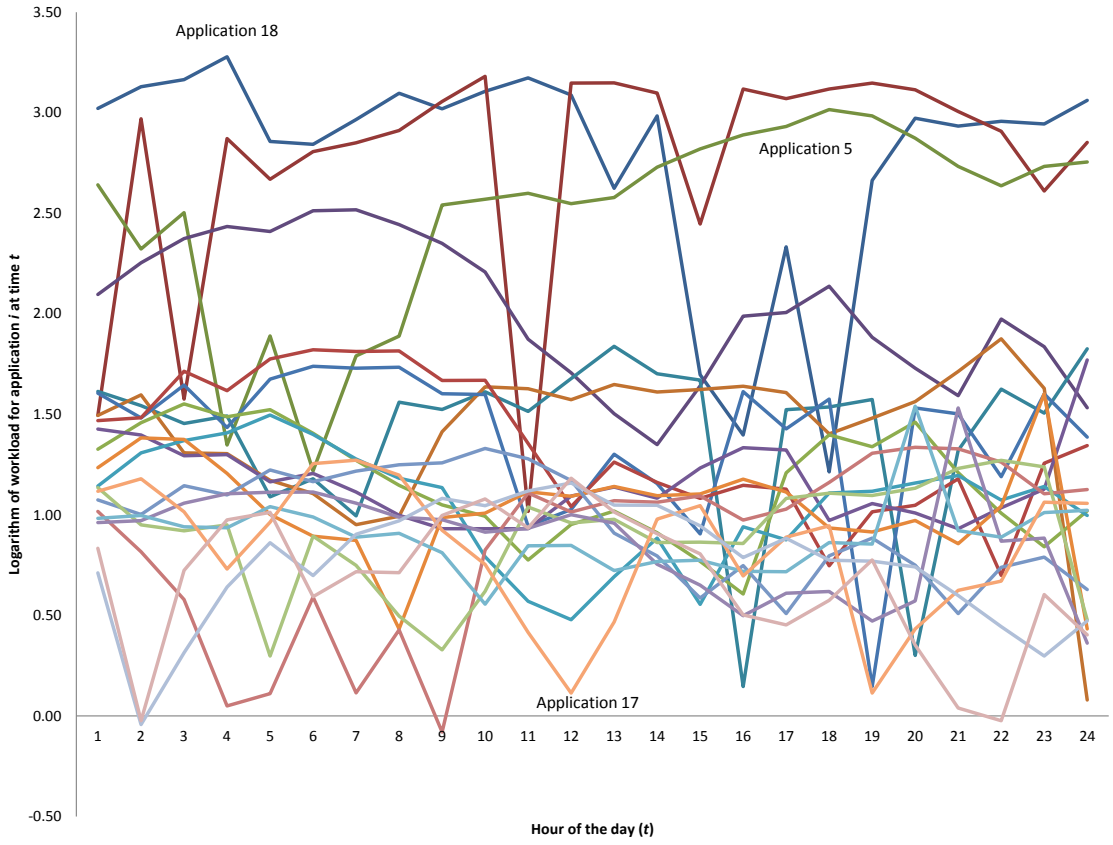

Figure 7: Real traces of hourly workloads for 20 applications for one day

i	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	...	$t=24$	C_i^2
1	552.5430	516.5070	474.4660	450.4430	390.3840	312.3070	366.3600	468.4600	...	486.4780	0.1084
2	10.6190	9.5780	10.8270	9.1610	7.2870	7.0790	8.5370	12.9090	...	13.1170	0.5516
3	101.7230	127.9480	148.6110	137.4850	149.4050	135.8950	152.5840	150.2000	...	88.2130	0.0848
4	51.5520	58.7120	67.3040	54.4160	60.1440	54.4160	55.8480	60.1440	...	61.5760	0.0738
5	101.1410	112.4590	97.6080	98.8000	122.4380	109.9750	91.2560	137.3370	...	104.4850	0.0206
6	7.7070	8.2890	8.5800	8.4340	9.8880	13.2330	17.3050	19.1950	...	6.3980	0.2448
7	7.1110	5.9680	5.1850	4.5740	4.2450	4.2760	4.3700	4.4800	...	6.4380	0.1058
8	0.0530	0.0520	0.0370	0.0270	0.0400	0.0220	0.0330	0.0670	...	0.0530	0.4796
9	5.0680	2.8680	1.5870	1.1700	0.8630	0.8080	1.1970	1.9770	...	6.7390	0.3035
10	3.7680	3.4540	2.8260	2.3550	2.1980	2.3550	2.9830	4.3960	...	5.0240	0.2476
11	1.8930	1.5610	1.3720	1.3250	1.4190	1.1830	1.1830	1.3250	...	1.8450	0.0990
12	2.0150	2.0620	2.0150	1.9210	2.2020	2.0620	2.2020	2.3900	...	1.8270	0.0328
13	0.1670	0.1800	0.2950	0.3460	0.4360	0.5000	0.5520	0.6670	...	3.4640	0.5947
14	20.8050	15.9600	13.9650	11.9700	8.2650	8.8350	9.4050	13.1100	...	20.8050	0.0966
15	8.7630	8.1250	9.2120	21.6370	21.1350	17.1590	18.8630	18.4370	...	8.4180	0.2147
16	0.3330	0.9990	0.3330	1.3320	0.9990	3.3300	3.3300	0.9990	...	0.6660	0.4962
17	27.5300	26.7930	19.9100	10.3240	7.1280	8.1120	8.6030	10.3240	...	24.8260	0.1956
18	11.8890	13.7800	14.5910	15.4010	16.2120	22.1560	27.2900	35.9360	...	11.6180	0.2289
19	10.1660	10.8210	11.8050	10.4930	19.0190	31.1520	28.5290	22.2990	...	3.6070	0.3343
20	11.3700	13.0860	15.4460	17.3770	18.6640	6.8650	21.0240	21.4530	...	7.5090	0.0985
C_t^2	6.8844	6.1308	5.6870	5.6832	4.7343	4.0473	4.3100	4.7485	...	6.3182	

Table 9: Sample of hourly workload traces for the 20 applications with C_i^2 and C_t^2

Next we present a sample of the synthetic traces we generated for our numerical experiments. In particular, this is an instance with high variability across applications and low variability across time (hence HL) which is similar to the real trace. These are referred to in Section 6.3 and is generated randomly using a geometric Brownian motion with parameters randomly sampled from a Pareto distribution. They are illustrated in Figure 8.

Next we present a sample of the synthetic traces we generated for our numerical experiments. These are referred to in Section 6.3 and were generated randomly using a geometric Brownian motion with parameters randomly sampled from a Pareto distribution. The traces are illustrated in Figure 8 and are similar to the real traces in that they have high variability across applications and low variability across time, hence HL. The traces in Figure 8 were generated by maintaining similar characteristics (such as C_i^2 and C_t^2) as in the real traces. From Figures 7 and 8, it is clear that our generated instances resemble the real traces.

i	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
1	1.0000	0.2793	-0.8112	0.6205	0.4572	-0.7654	0.7694	0.0529	0.6299	0.3920
2	0.2793	1.0000	0.0653	0.7359	0.5091	0.0490	0.4716	0.9046	0.8126	0.9622
3	-0.8112	0.0653	1.0000	-0.4094	-0.1452	0.8474	-0.5055	0.2597	-0.4730	-0.0546
4	0.6205	0.7359	-0.4094	1.0000	0.6027	-0.3156	0.5819	0.5831	0.8218	0.7845
5	0.4572	0.5091	-0.1452	0.6027	1.0000	-0.0761	0.4443	0.3282	0.4485	0.5264
6	-0.7654	0.0490	0.8474	-0.3156	-0.0761	1.0000	-0.5937	0.2629	-0.3564	-0.0096
7	0.7694	0.4716	-0.5055	0.5819	0.4443	-0.5937	1.0000	0.3080	0.6323	0.5540
8	0.0529	0.9046	0.2597	0.5831	0.3282	0.2629	0.3080	1.0000	0.6861	0.8790
9	0.6299	0.8126	-0.4730	0.8218	0.4485	-0.3564	0.6323	0.6861	1.0000	0.8752
10	0.3920	0.9622	-0.0546	0.7845	0.5264	-0.0096	0.5540	0.8790	0.8752	1.0000
11	0.5618	0.8873	-0.2751	0.7507	0.5784	-0.3030	0.6604	0.6769	0.8962	0.8908
12	-0.6919	0.2286	0.9081	-0.2928	0.0662	0.8456	-0.3811	0.3871	-0.2583	0.1519
13	0.6105	0.4864	-0.5618	0.7951	0.4859	-0.3471	0.4467	0.4038	0.7547	0.5701
14	0.7668	0.3156	-0.8010	0.5958	0.2617	-0.6435	0.5669	0.1923	0.7661	0.4031
15	-0.5584	-0.3769	0.6246	-0.6195	-0.1968	0.5117	-0.4709	-0.2985	-0.7132	-0.4422
16	-0.1699	-0.1040	0.0489	-0.0598	0.0064	0.2103	-0.2873	-0.2617	-0.1932	-0.1786
17	0.2862	-0.3021	-0.5878	0.0286	-0.2415	-0.5329	0.1666	-0.2581	0.1397	-0.2304
18	-0.5103	0.3702	0.7508	0.0251	0.2815	0.8702	-0.2803	0.4497	-0.0571	0.3403
19	-0.6745	0.1729	0.8368	-0.2469	0.0038	0.8410	-0.3638	0.3080	-0.3340	0.0553
20	-0.3907	0.2872	0.6956	-0.0930	0.0328	0.6215	-0.2389	0.4428	-0.1210	0.1656
i	$k=11$	$k=12$	$k=13$	$k=14$	$k=15$	$k=16$	$k=17$	$k=18$	$k=19$	$k=20$
1	0.5618	-0.6919	0.6105	0.7668	-0.5584	-0.1699	0.2862	-0.5103	-0.6745	-0.3907
2	0.8873	0.2286	0.4864	0.3156	-0.3769	-0.1040	-0.3021	0.3702	0.1729	0.2872
3	-0.2751	0.9081	-0.5618	-0.8010	0.6246	0.0489	-0.5878	0.7508	0.8368	0.6956
4	0.7507	-0.2928	0.7951	0.5958	-0.6195	-0.0598	0.0286	0.0251	-0.2469	-0.0930
5	0.5784	0.0662	0.4859	0.2617	-0.1968	0.0064	-0.2415	0.2815	0.0038	0.0328
6	-0.3030	0.8456	-0.3471	-0.6435	0.5117	0.2103	-0.5329	0.8702	0.8410	0.6215
7	0.6604	-0.3811	0.4467	0.5669	-0.4709	-0.2873	0.1666	-0.2803	-0.3638	-0.2389
8	0.6769	0.3871	0.4038	0.1923	-0.2985	-0.2617	-0.2581	0.4497	0.3080	0.4428
9	0.8962	-0.2583	0.7547	0.7661	-0.7132	-0.1932	0.1397	-0.0571	-0.3340	-0.1210
10	0.8908	0.1519	0.5701	0.4031	-0.4422	-0.1786	-0.2304	0.3403	0.0553	0.1656
11	1.0000	-0.0513	0.5235	0.5485	-0.4965	-0.1592	-0.1162	0.0719	-0.1682	0.0265
12	-0.0513	1.0000	-0.4787	-0.6795	0.4938	-0.0103	-0.6086	0.8238	0.7832	0.6417
13	0.5235	-0.4787	1.0000	0.7165	-0.6773	-0.0650	0.2687	-0.1169	-0.3568	-0.2730
14	0.5485	-0.6795	0.7165	1.0000	-0.8010	-0.1601	0.5880	-0.4812	-0.6972	-0.3673
15	-0.4965	0.4938	-0.6773	-0.8010	1.0000	0.2601	-0.6789	0.3784	0.5552	0.4271
16	-0.1592	-0.0103	-0.0650	-0.1601	0.2601	1.0000	-0.3121	0.1997	0.3495	-0.0946
17	-0.1162	-0.6086	0.2687	0.5880	-0.6789	-0.3121	1.0000	-0.6155	-0.6624	-0.5267
18	0.0719	0.8238	-0.1169	-0.4812	0.3784	0.1997	-0.6155	1.0000	0.7742	0.5362
19	-0.1682	0.7832	-0.3568	-0.6972	0.5552	0.3495	-0.6624	0.7742	1.0000	0.5741
20	0.0265	0.6417	-0.2730	-0.3673	0.4271	-0.0946	-0.5267	0.5362	0.5741	1.0000

Table 10: $C_r(i,k)$ values for the real traces

Acknowledgments

This research was partially supported by the NSF grant CMMI-0946935. The authors would like to thank the editors and the anonymous referees for their comments which resulted in a significant improvement in the presentation of this paper.

References

[1] L. Bertini, J. Leite, and D. Mossé. Dynamic configuration of web server clusters with QoS control. In *WIP Session of the 20th Euromicro Conference on Real-Time Systems*, 2008.

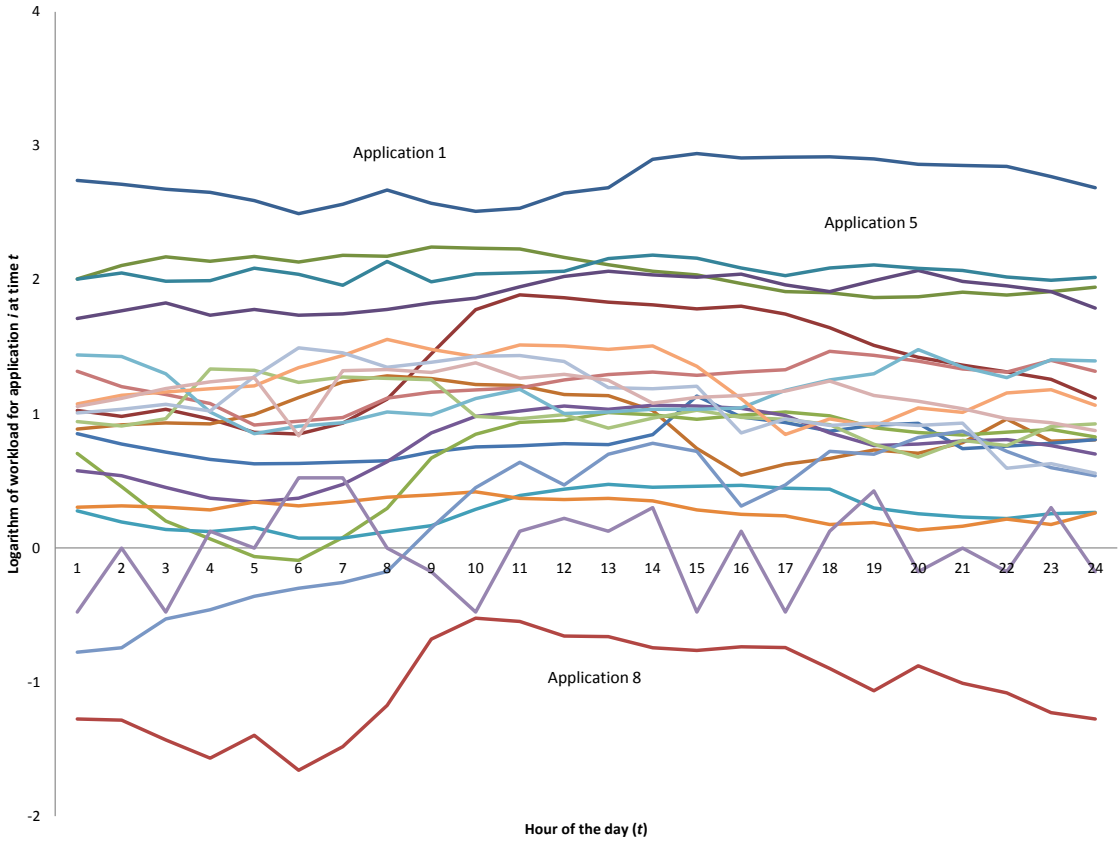

Figure 8: Synthetically generated hourly workloads for 20 applications for one day

[2] A. Chandra, P. Goyal, and P. Shenoy. Quantifying the benefits of resource multiplexing in on-demand data centers. In *Proceedings of First ACM Workshop on Algorithms and Architectures for Self-Managing Systems*, 2003.

[3] H. Chen, J. Marden, and A. Wierman. On the impact of heterogeneity and back-end scheduling in load balancing designs. In *Proceedings of INFOCOM*, 2009.

[4] H. Chen and D.D. Yao. *Fundamentals of Queueing Networks*. Springer-Verlag, New York, NY, 2001.

[5] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. *ACM SIGMETRICS Performance Evaluation Review*, 33(1):303–314, 2005.

[6] W.W. Chu and L.M.-T. Lan. Task allocation and precedence relations for distributed real-time systems. *IEEE Transactions on Computers*, C-36(6):667–679, june 1987.

[7] McKinsey & Company. Revolutionizing data center efficiency, April 30 2008. Uptime Institute Symposium http://uptimeinstitute.org/content/view/168/57.

[8] Adena DeMonte. Why we need green data centers. http://earth2tech.com/2007/08/02/why-we-need-green-data-centers/.

[9] G. Dhiman, K.K. Pusukuri, and T. Rosing. Analysis of dynamic voltage scaling for system level energy management. In *Proceedings of the USENIX Workshop on Power-Aware Computing and Systems (HotPower)*, 2008.

[10] Darrell Dunn. Stayin' alive: Living happily with your existing data center, September 20 2007. Computer World http://www.computerworld.com/action/article.do?command=view ArticleBasic&articleId=9037978&pageNumber=1.

[11] Vineeta Durani. IBM unveils plan to combat data center energy crisis, May 10 2007. http://www-03.ibm.com/press/us/en/pressrelease/21524.wss.

[12] J.J.G. Garcia and M.G. Harbour. Optimized priority assignment for tasks and messages in distributed hard real-time systems. In *Proceedings of the Third Workshop on Parallel and Distributed Real-Time Systems*, pages 124–132, Apr 1995.

[13] S. Herbert and D. Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design*, page 43. ACM, 2007.

[14] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Transactions on Computers*, 56(4):444–458, 2004.

[15] C.E. Houstis. Module allocation of real-time applications to distributed systems. *Software Engineering, IEEE Transactions on*, 16(7):699–709, jul 1990.

[16] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15, 2009.

[17] M LaMonica. Mckinsey: Servers need an mpg-like energy rating. http://news.cnet.com/8301-11128_3-9932184-54.html.

[18] K. Le, R. Bianchini, M. Martonosi, and T. Nguyen. Cost-and energy-aware load distribution across data centers. In *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower)*, 2009.

[19] S. Mandel. Rooms that consume internet hotels and other data centers inhale electricity. What's the problem with that? *Electric Perspectives*, 26(3):34–51, 2001.

[20] D. Meisner, B.T. Gold, and T.F. Wenisch. Powernap: eliminating server idle power. *ACM SIGPLAN Notices*, 44(3):205–216, 2009.

[21] Robert Mitchell. Power struggle: How it managers cope with the data center power demands, April 03 2006. `Computer World http://www.computerworld.com/hardwaretopics/hardware/story/0,10801,110072,00.html`.

[22] P. Padala, K.G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive control of virtualized resources in utility computing environments. *ACM SIGOPS Operating Systems Review*, 41(3):289–302, 2007.

[23] C.D. Patel, R. Sharma, C.E. Bash, and A. Beitelmal. Thermal considerations in cooling large scale high compute density data centers. In *8th ITHERM Conference, San Diego*, 2002.

[24] S. Pelley, D. Meisner, T.F. Wenisch, and J.W. VanGilder. Understanding and abstracting total data center power. In *WEED: Workshop on Energy Efficienct Design*, 2009.

[25] V. Petrucci, O. Loques, and D. Mossé. A dynamic configuration model for power-efficient virtualized server clusters. In *11th Brazillian Workshop on Real-Time and Embedded Systems (WTR)*, 2009.

[26] A. Singla, C. Hong, L. Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers, randomly. In *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2011.

[27] Patrick Thibodeau. EPA: U.S. needs more power plants to support data centers, August 03 2007. `Computer World http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9028958`.

[28] K. W. Tindell, A. Burns, and A. J. Wellings. Allocating hard real-time tasks: An NP-hard problem made easy. *Real-Time Systems*, 4:145–165, 1992.

[29] B. Urgaonkar, A. Rosenberg, and P. Shenoy. Application placement on a cluster of servers. *Proceedings of First ACM Workshop on Algorithms and Architectures for Self-Managing Systems, June 2003*, 18(5):1023–1041, oct 2007.

[30] A. Verma, G. Dasgupta, T.K. Nayak, P. De, and R. Kothari. Server workload analysis for power minimization using consolidation. In *Proceedings of USENIX Annual Technical Conference*, 2009.

[31] Y. Wang, X. Wang, M. Chen, and X. Zhu. Power-efficient response time guarantees for virtualized enterprise servers. In *Real-Time Systems Symposium, 2008*, pages 303–312, 2008.

[32] W. Whitt. *Stochastic-Process Limits*. Springer-Verlag, New York, NY, 2002.